KU LEUVEN

Fully Abstract Compilation from System F to λ^{σ}

Dominique Devriese[1], Marco Patrignani[2], Frank Piessens[1]

[1] iMinds-DistriNet, KU Leuven,
[2] MPI-SWS

Paris, 2016

System F and λ^σ

Compiling System F to λ^σ

Proving Full Abstraction

Constructing a back-translation

$$\tau ::= \texttt{Unit} \mid \texttt{Bool} \mid \tau \to \tau \mid \tau \times \tau \mid \tau \uplus \tau \mid \forall\alpha.\,\tau \mid \exists\alpha.\,\tau \mid \mu\alpha.\,\tau \mid \alpha$$

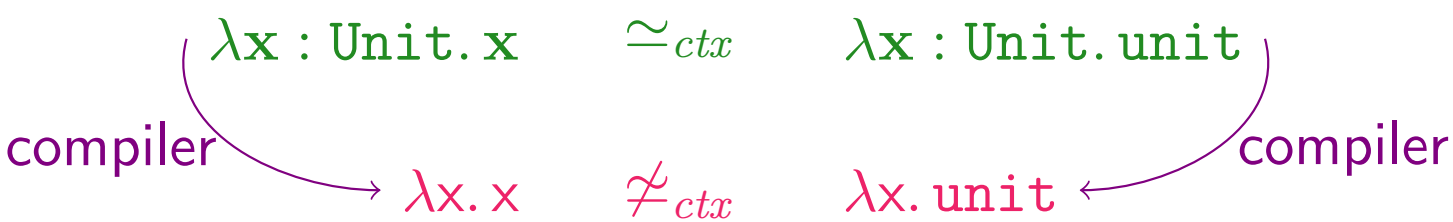
$$\begin{aligned}\mathbf{t} ::= {} & \texttt{unit} \mid \texttt{true} \mid \texttt{false} \mid \lambda\mathbf{x}:\tau.\,\mathbf{t} \mid \mathbf{x} \mid \mathbf{t}\ \mathbf{t} \mid \mathbf{t.1} \mid \mathbf{t.2} \mid \langle \mathbf{t}, \mathbf{t}\rangle \mid \mathbf{\Lambda}\alpha.\,\mathbf{t} \\ & \mid \mathbf{t}\ \tau \mid \text{inl}\ \mathbf{t} \mid \text{inr}\ \mathbf{t} \mid \text{case}\ \mathbf{t}\ \text{of inl}\ \mathbf{x_1} \mapsto \mathbf{t} \mid \text{inr}\ \mathbf{x_2} \mapsto \mathbf{t} \mid \mathbf{t};\mathbf{t} \\ & \mid \text{if}\ \mathbf{t}\ \text{then}\ \mathbf{t}\ \text{else}\ \mathbf{t} \mid \text{pack}\ \langle \tau, \mathbf{t}\rangle\ \text{as}\ \exists\alpha.\,\tau \\ & \mid \text{unpack}\ \mathbf{t}\ \text{as}\ \langle \alpha, \mathbf{x}\rangle\ \text{in}\ \mathbf{t} \mid \text{roll}\ \mathbf{t} \mid \text{unroll}\ \mathbf{t}\end{aligned}$$

$$\mathsf{t} ::= \mathtt{unit} \mid \mathtt{true} \mid \mathtt{false} \mid \lambda \mathsf{x}.\,\mathsf{t} \mid \mathsf{x} \mid \mathsf{t}\ \mathsf{t} \mid \mathsf{t}.1 \mid \mathsf{t}.2 \mid \langle \mathsf{t}, \mathsf{t} \rangle \mid \mathrm{inl}\ \mathsf{t} \mid \mathrm{inr}\ \mathsf{t}$$
$$\mid \mathrm{case}\ \mathsf{t}\ \mathrm{of}\ \mathrm{inl}\ \mathsf{x}_1 \mapsto \mathsf{t} \mid \mathrm{inr}\ \mathsf{x}_2 \mapsto \mathsf{t} \mid \mathsf{t}; \mathsf{t} \mid \mathrm{if}\ \mathsf{t}\ \mathrm{then}\ \mathsf{t}\ \mathrm{else}\ \mathsf{t}$$
$$\mid \mathsf{wrong} \mid \nu \mathsf{x}.\mathsf{t} \mid \{\mathsf{t}\}_\mathsf{t} \mid \sigma \mid \mathrm{let}\ \{\mathsf{x}\}_\mathsf{t} = \mathsf{t}\ \mathrm{in}\ \mathsf{t}\ \mathrm{else}\ \mathsf{t}$$

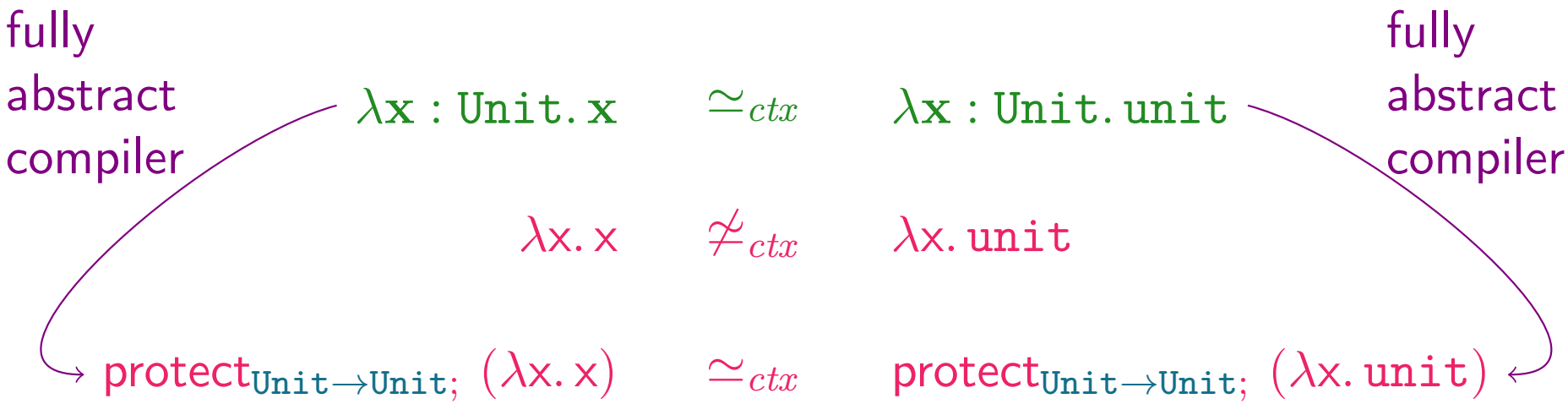
$$\frac{\sigma \notin \mathrm{dom}(\mathsf{h})}{(\mathsf{h}, \nu \mathsf{x}.\mathsf{t}) \hookrightarrow (\mathsf{h}; \sigma, \mathsf{t}[\sigma/\mathsf{x}])}$$

$$\frac{\sigma \equiv \sigma'}{\mathrm{let}\ \{\mathsf{x}\}_\sigma = \{\mathsf{v}\}_{\sigma'}\ \mathrm{in}\ \mathsf{t}\ \mathrm{else}\ \mathsf{t}' \hookrightarrow \mathsf{t}[\mathsf{v}/\mathsf{x}]}$$

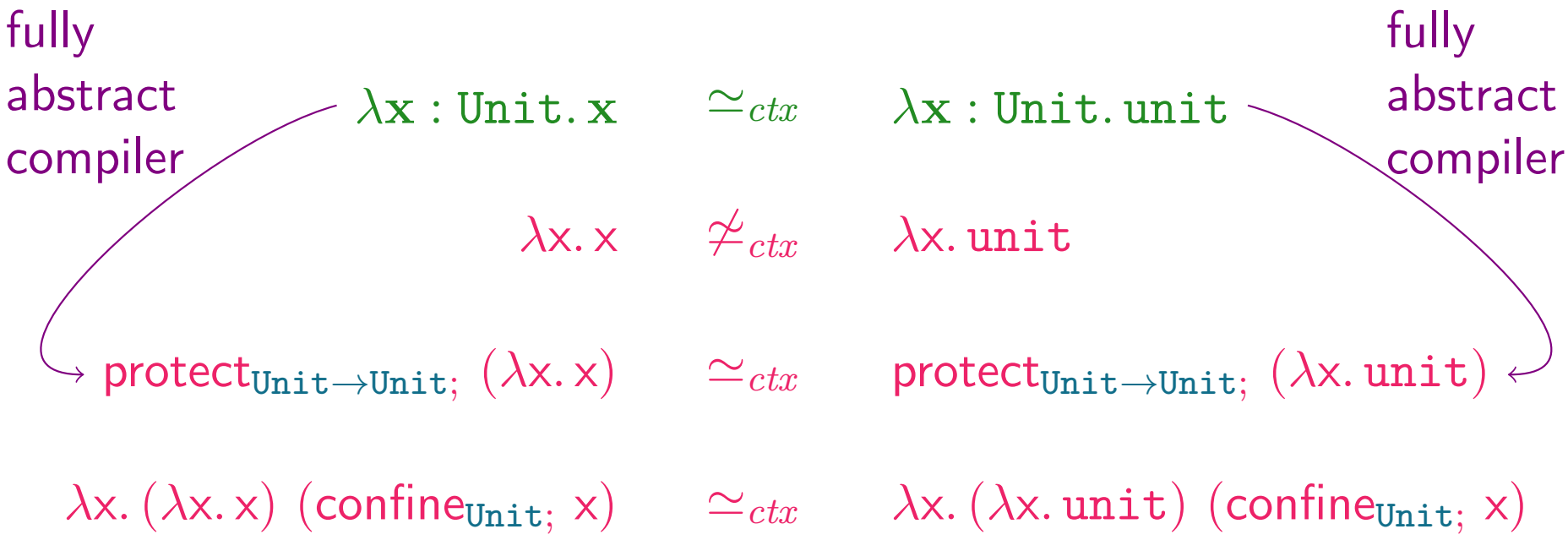
$$\frac{\sigma \not\equiv \sigma'}{\mathrm{let}\ \{\mathsf{x}\}_\sigma = \{\mathsf{v}\}_{\sigma'}\ \mathrm{in}\ \mathsf{t}\ \mathrm{else}\ \mathsf{t}' \hookrightarrow \mathsf{t}'}$$

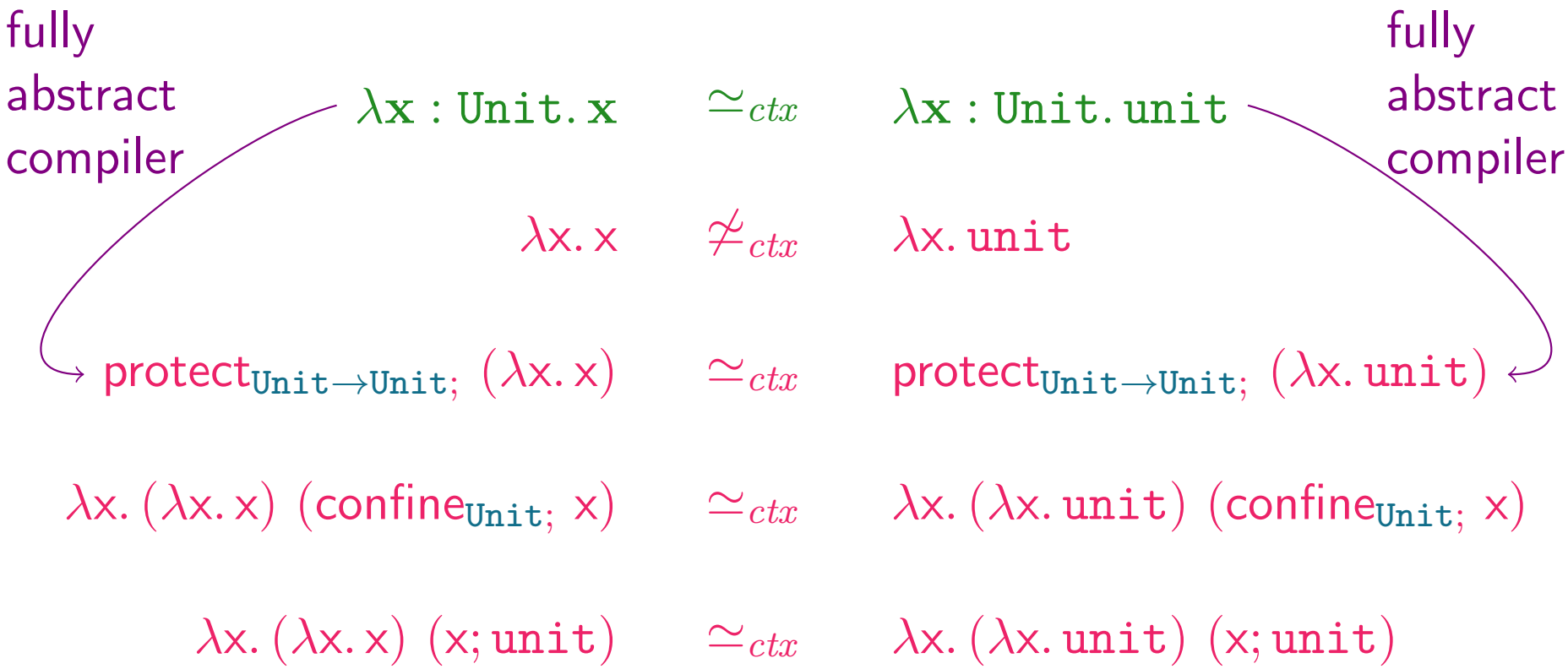

$\lambda \mathtt{x} : \mathtt{Unit}.\, \mathtt{x}$
$\simeq_{ctx}$
$\lambda \mathtt{x} : \mathtt{Unit}.\, \mathtt{unit}$
compiler
compiler
$\lambda \mathtt{x}.\, \mathtt{x}$
$\not\simeq_{ctx}$
$\lambda \mathtt{x}.\, \mathtt{unit}$

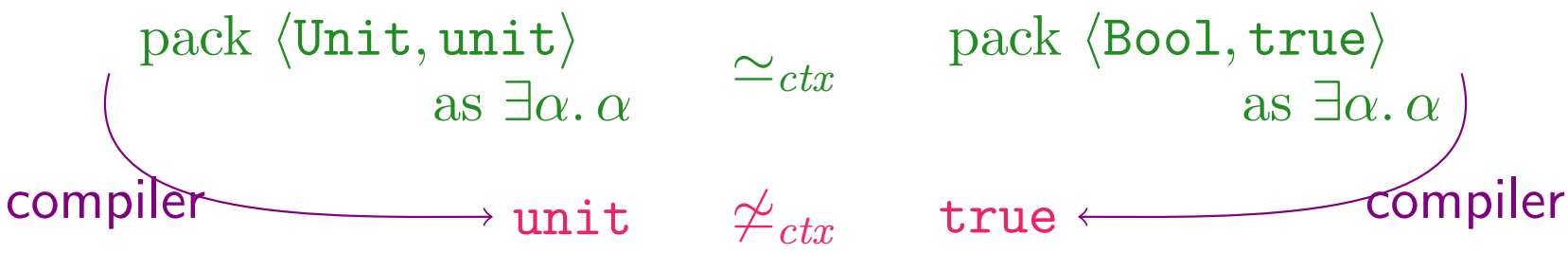
KU LEUVEN

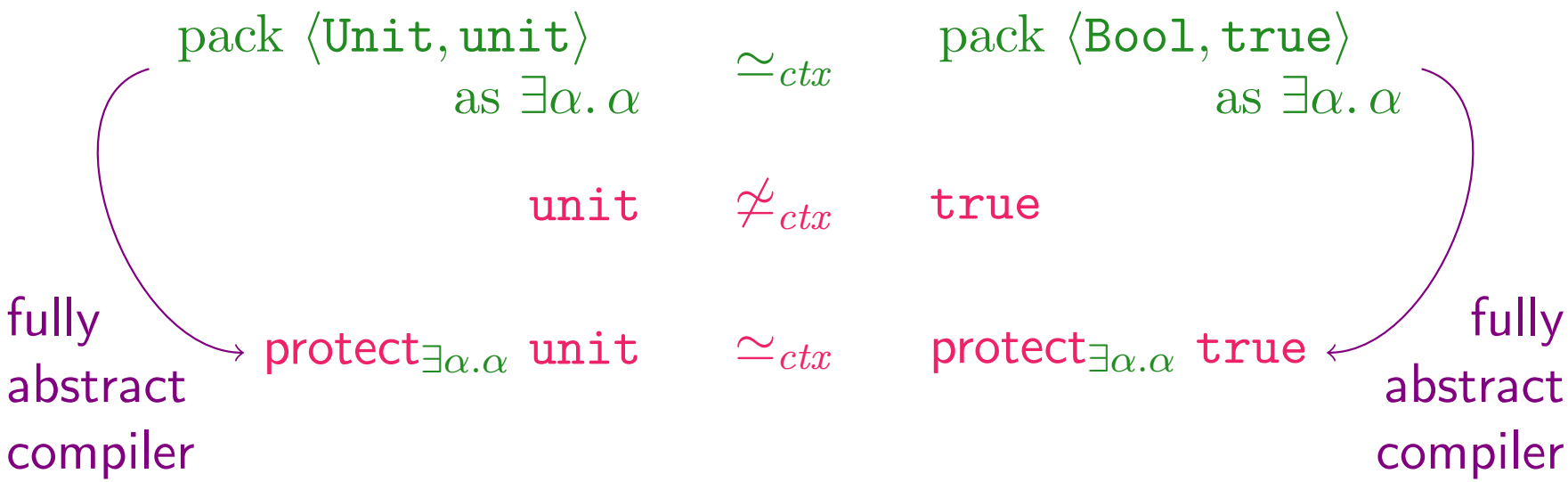

fully abstract compiler
$\lambda \mathtt{x} : \mathtt{Unit}.\, \mathtt{x}$ $\simeq_{ctx}$ $\lambda \mathtt{x} : \mathtt{Unit}.\, \mathtt{unit}$
$\lambda \mathtt{x}.\, \mathtt{x}$ $\not\simeq_{ctx}$ $\lambda \mathtt{x}.\, \mathtt{unit}$
$\mathsf{protect}_{\mathtt{Unit}\to\mathtt{Unit};}\ (\lambda \mathtt{x}.\, \mathtt{x})$ $\simeq_{ctx}$ $\mathsf{protect}_{\mathtt{Unit}\to\mathtt{Unit};}\ (\lambda \mathtt{x}.\, \mathtt{unit})$
fully abstract compiler

KU LEUVEN

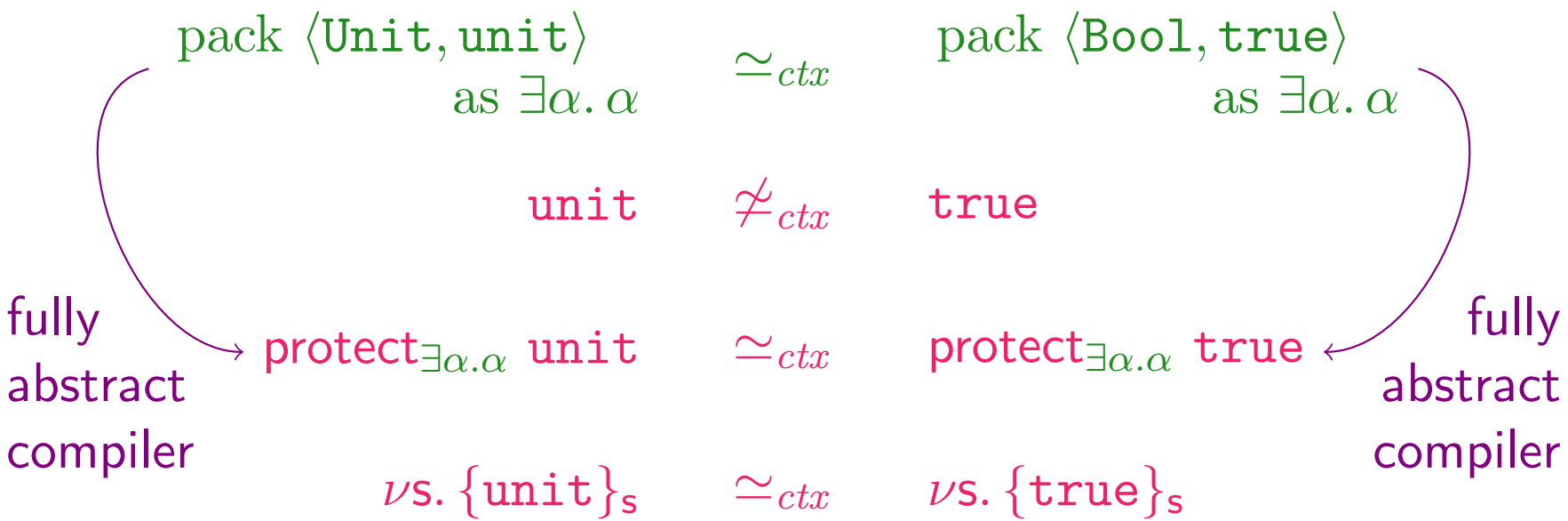

fully abstract compiler
fully abstract compiler
$\lambda \mathtt{x} : \mathtt{Unit}.\, \mathtt{x}$ $\simeq_{ctx}$ $\lambda \mathtt{x} : \mathtt{Unit}.\, \mathtt{unit}$
$\lambda \mathtt{x}.\, \mathtt{x}$ $\not\simeq_{ctx}$ $\lambda \mathtt{x}.\, \mathtt{unit}$
$\mathsf{protect}_{\mathtt{Unit}\to\mathtt{Unit};}\ (\lambda \mathtt{x}.\, \mathtt{x})$ $\simeq_{ctx}$ $\mathsf{protect}_{\mathtt{Unit}\to\mathtt{Unit};}\ (\lambda \mathtt{x}.\, \mathtt{unit})$
$\lambda \mathtt{x}.\, (\lambda \mathtt{x}.\, \mathtt{x})\ (\mathsf{confine}_{\mathtt{Unit};}\ \mathtt{x})$ $\simeq_{ctx}$ $\lambda \mathtt{x}.\, (\lambda \mathtt{x}.\, \mathtt{unit})\ (\mathsf{confine}_{\mathtt{Unit};}\ \mathtt{x})$


fully abstract compiler
fully abstract compiler
$\lambda \mathtt{x} : \mathtt{Unit}.\, \mathtt{x} \simeq_{ctx} \lambda \mathtt{x} : \mathtt{Unit}.\, \mathtt{unit}$
$\lambda \mathtt{x}.\, \mathtt{x} \not\simeq_{ctx} \lambda \mathtt{x}.\, \mathtt{unit}$
$\mathsf{protect}_{\mathtt{Unit}\to\mathtt{Unit};}\ (\lambda \mathtt{x}.\, \mathtt{x}) \simeq_{ctx} \mathsf{protect}_{\mathtt{Unit}\to\mathtt{Unit};}\ (\lambda \mathtt{x}.\, \mathtt{unit})$
$\lambda \mathtt{x}.\, (\lambda \mathtt{x}.\, \mathtt{x})\ (\mathsf{confine}_{\mathtt{Unit};}\ \mathtt{x}) \simeq_{ctx} \lambda \mathtt{x}.\, (\lambda \mathtt{x}.\, \mathtt{unit})\ (\mathsf{confine}_{\mathtt{Unit};}\ \mathtt{x})$
$\lambda \mathtt{x}.\, (\lambda \mathtt{x}.\, \mathtt{x})\ (\mathtt{x}; \mathtt{unit}) \simeq_{ctx} \lambda \mathtt{x}.\, (\lambda \mathtt{x}.\, \mathtt{unit})\ (\mathtt{x}; \mathtt{unit})$


pack ⟨Unit, unit⟩
as ∃α. α
≃ctx
pack ⟨Bool, true⟩
as ∃α. α
compiler
unit
≄ctx
true
compiler


pack ⟨Unit, unit⟩ as ∃α. α
≃ctx
pack ⟨Bool, true⟩ as ∃α. α
unit
≄ctx
true
fully abstract compiler
protect∃α.α unit
≃ctx
protect∃α.α true
fully abstract compiler

KU LEUVEN


pack ⟨Unit, unit⟩ as ∃α. α
≃ctx
pack ⟨Bool, true⟩ as ∃α. α
unit
≄ctx
true
fully abstract compiler
protect∃α.α unit
≃ctx
protect∃α.α true
fully abstract compiler
νs. {unit}s
≃ctx
νs. {true}s

KU LEUVEN

More interesting examples:

$$\mathbf{t} \stackrel{\text{def}}{=} \mathbf{pack}\ \langle \mathtt{Unit}, \langle \mathtt{unit}, \lambda x : \mathtt{Unit}.\, x \rangle \rangle \ \mathbf{as}\ \exists \alpha.\, \alpha \times (\alpha \to \mathtt{Unit})$$

$$\mathtt{erase}(\mathbf{t}) = \langle \mathtt{unit}, \lambda \mathtt{x}.\, \mathtt{x} \rangle$$

$$\mathsf{protect}_{\exists \alpha.\alpha \times (\alpha \to \mathtt{Unit})}\ \mathtt{erase}(\mathbf{t}) = \nu \mathtt{s}.\, \langle \{\mathtt{unit}\}_{\mathtt{s}}, \lambda \mathtt{x}.\, \mathsf{unseal}_{\mathtt{s}}\ \mathtt{x} \rangle$$

$$\mathbf{t} \stackrel{\text{def}}{=} \lambda \mathbf{f} : \forall \alpha.\, \alpha \to \alpha.\, \mathbf{f}\ \mathtt{Bool}\ \mathtt{true}$$

$$\mathtt{erase}(\mathbf{t}) = \lambda \mathtt{f}.\, \mathtt{f}\ \mathtt{unit}\ \mathtt{true}$$

$$\mathsf{protect}_{(\forall \alpha.\alpha \to \alpha) \to \mathtt{Bool}}\ \mathtt{erase}(\mathbf{t}) = \lambda \mathtt{f}.\, \nu \mathtt{s}.\, \mathsf{unseal}_{\mathtt{s}}\ (\mathtt{f}\ \mathtt{unit}\ \{\mathtt{true}\}_{\mathtt{s}})$$

$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$ iff $[\![\mathbf{t_1}]\!] \simeq_{ctx} [\![\mathbf{t_2}]\!]$

- Two directions: Contextual equivalence reflection and preservation
- Preservation is hardest to prove

Contextual Equivalence Reflection (1/2).
Prove using *cross-language logical relations*.

$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$[\![\mathbf{t_1}]\!] \simeq_{ctx} [\![\mathbf{t_2}]\!]$$

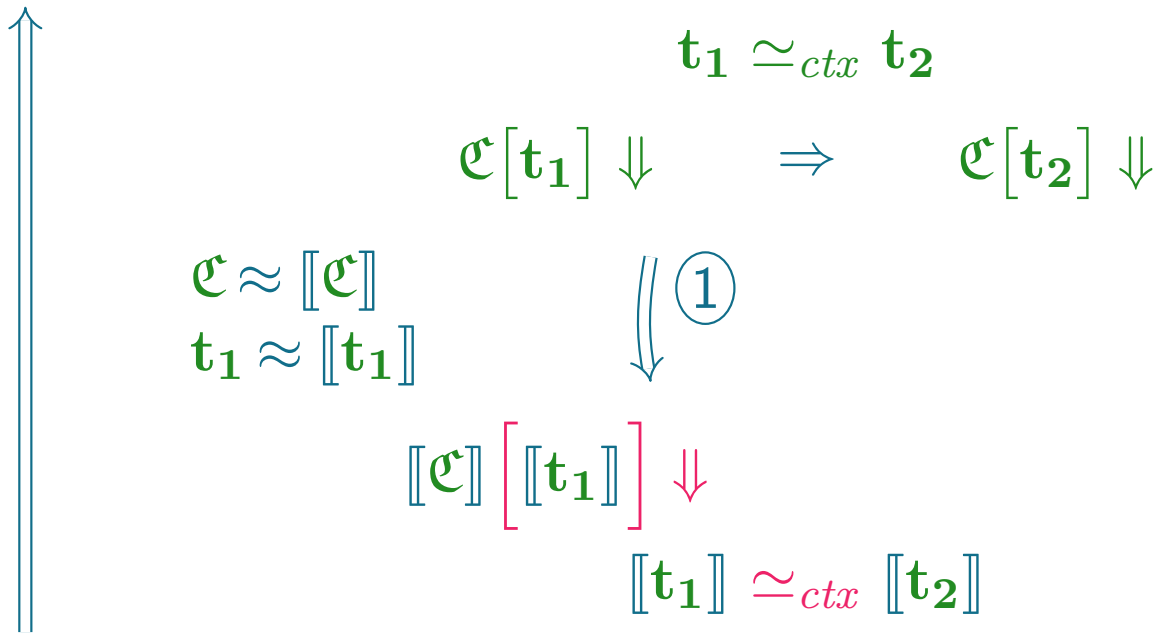
Contextual Equivalence Reflection (1/2).
Prove using *cross-language logical relations*.

$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$\mathfrak{C}[\mathbf{t_1}] \Downarrow \quad \Rightarrow \quad \mathfrak{C}[\mathbf{t_2}] \Downarrow$$

$$[\![\mathbf{t_1}]\!] \simeq_{ctx} [\![\mathbf{t_2}]\!]$$

Contextual Equivalence Reflection (1/2).
Prove using *cross-language logical relations*.



Contextual Equivalence Reflection (1/2).
Prove using *cross-language logical relations*.

$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

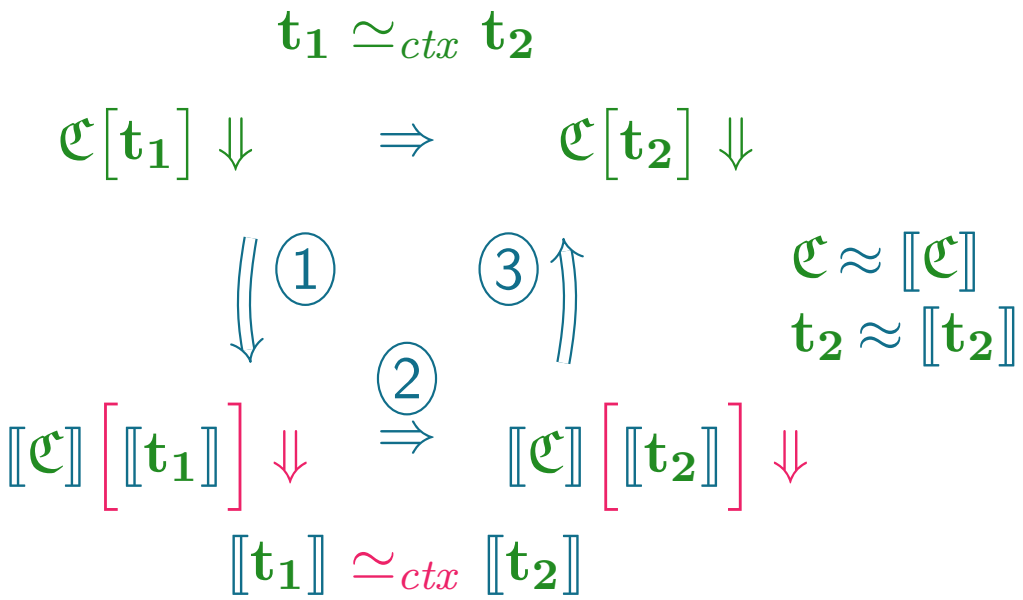
$$\mathfrak{C}[\mathbf{t_1}] \Downarrow \quad \Rightarrow \quad \mathfrak{C}[\mathbf{t_2}] \Downarrow$$

$$\Downarrow \ (1)$$

$$[\![\mathfrak{C}]\!]\left[[\![\mathbf{t_1}]\!]\right] \Downarrow \quad (2)\ \Rightarrow \quad [\![\mathfrak{C}]\!]\left[[\![\mathbf{t_2}]\!]\right] \Downarrow$$

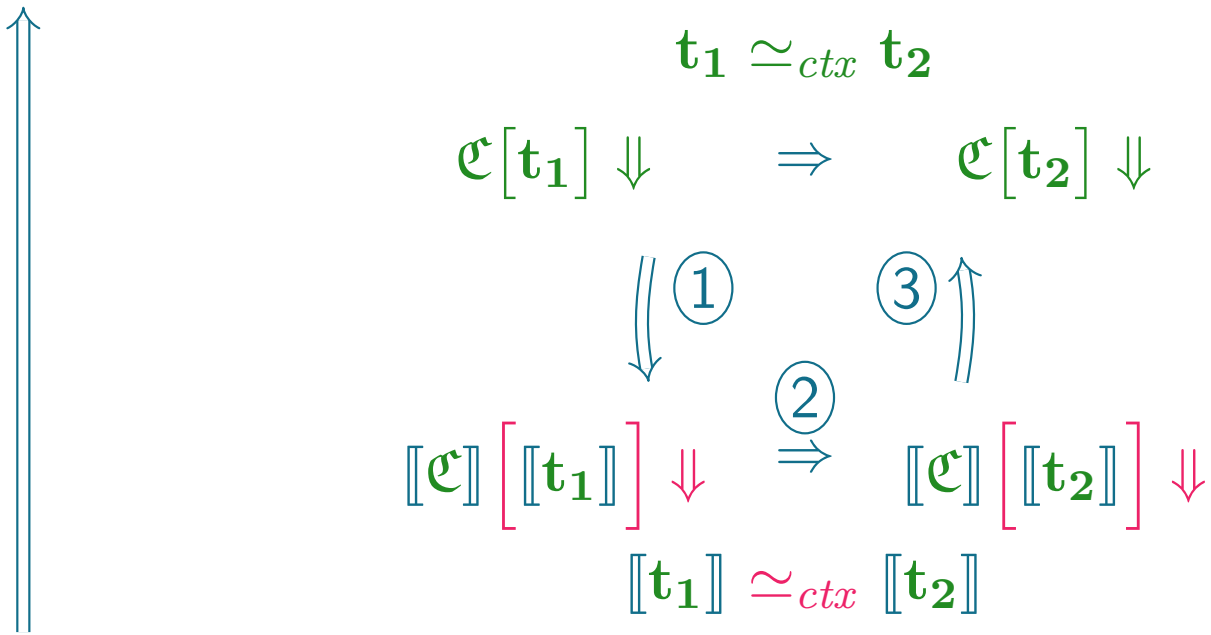
$$[\![\mathbf{t_1}]\!] \simeq_{ctx} [\![\mathbf{t_2}]\!]$$

Contextual Equivalence Reflection (1/2).
Prove using *cross-language logical relations*.



KU LEUVEN

Contextual Equivalence Reflection (1/2).
Prove using *cross-language logical relations*.



Contextual Equivalence Preservation (2/2).

$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$[\![\mathbf{t_1}]\!] \simeq_{ctx} [\![\mathbf{t_2}]\!]$$

Contextual Equivalence Preservation (2/2).

$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$\mathfrak{C}\left[\left[\mathbf{t_1}\right]\right] \Downarrow \quad \Rightarrow \quad \mathfrak{C}\left[\left[\mathbf{t_2}\right]\right] \Downarrow$$

$$\left[\left[\mathbf{t_1}\right]\right] \simeq_{ctx} \left[\left[\mathbf{t_2}\right]\right]$$

Contextual Equivalence Preservation (2/2).

Back-translation of target contexts: $\langle\langle \mathfrak{C} \rangle\rangle \approx \mathfrak{C}$

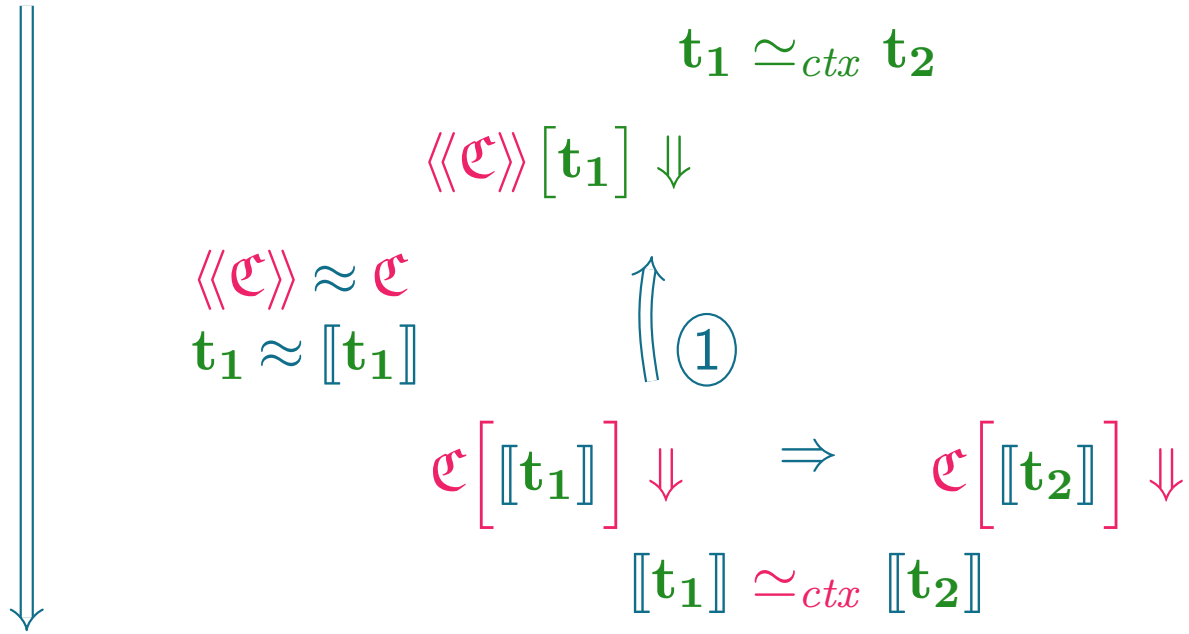
$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$\mathfrak{C}\left[\left[\mathbf{t_1}\right]\right] \Downarrow \quad \Rightarrow \quad \mathfrak{C}\left[\left[\mathbf{t_2}\right]\right] \Downarrow$$

$$\left[\left[\mathbf{t_1}\right]\right] \simeq_{ctx} \left[\left[\mathbf{t_2}\right]\right]$$

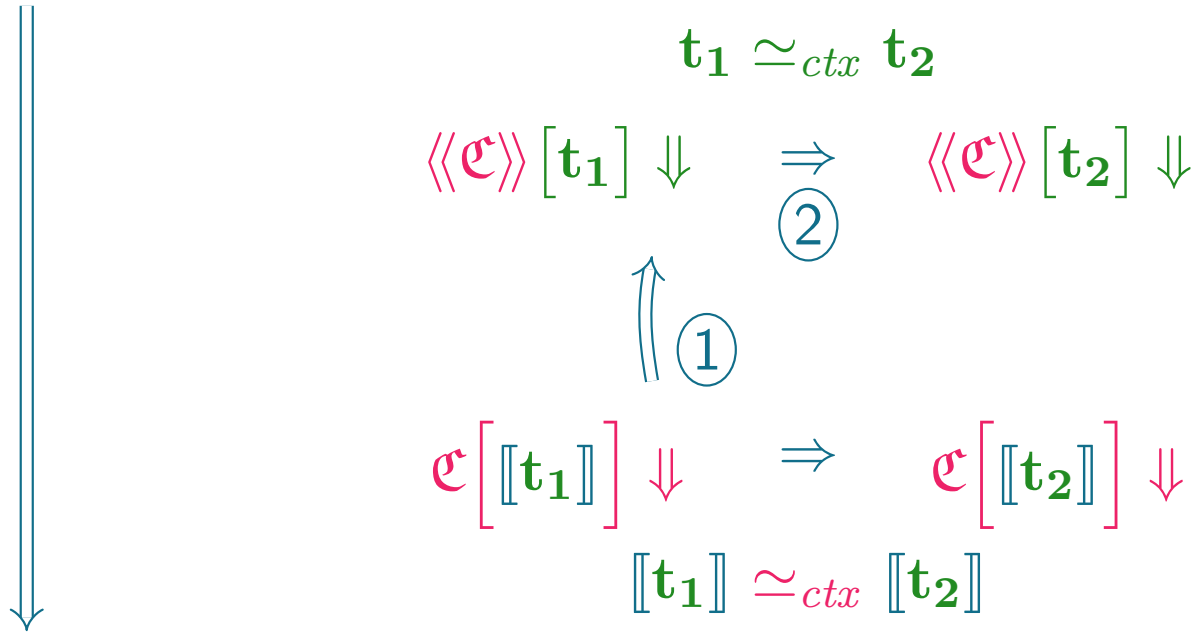
Contextual Equivalence Preservation (2/2).

Back-translation of target contexts: $\langle\!\langle \mathfrak{C} \rangle\!\rangle \approx \mathfrak{C}$



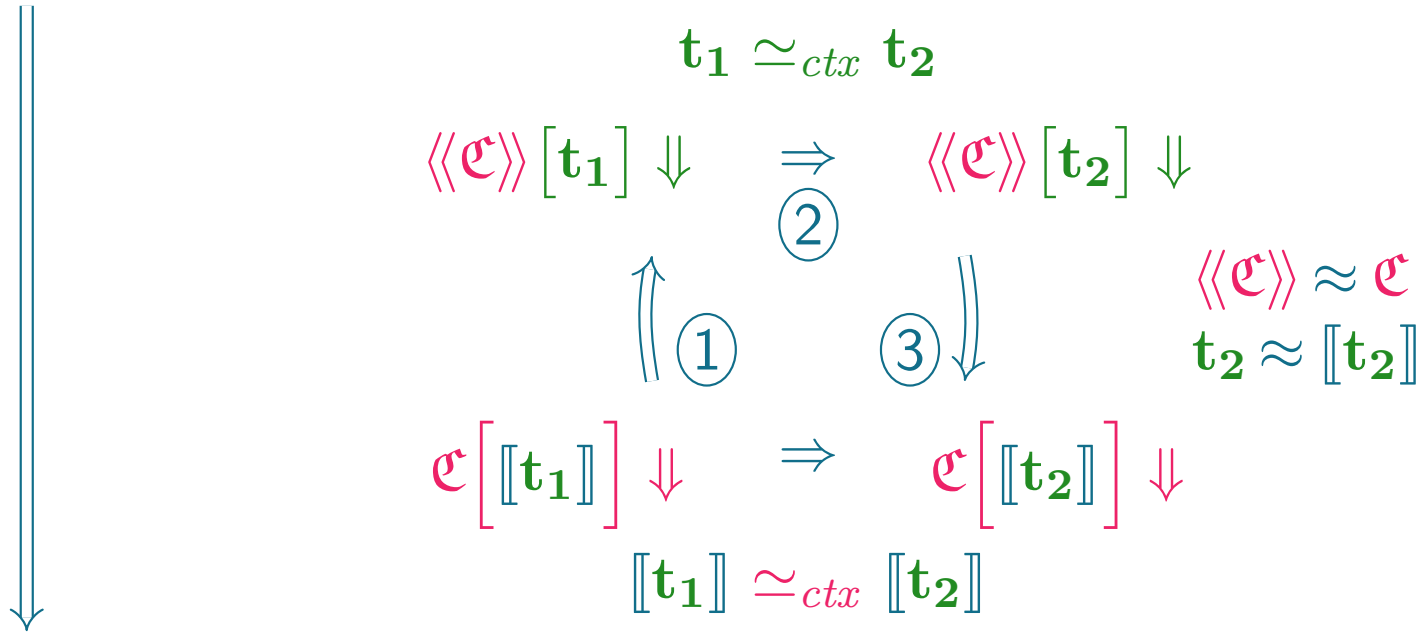
Contextual Equivalence Preservation (2/2).

Back-translation of target contexts: $\langle\!\langle \mathfrak{C} \rangle\!\rangle \approx \mathfrak{C}$



Contextual Equivalence Preservation (2/2).

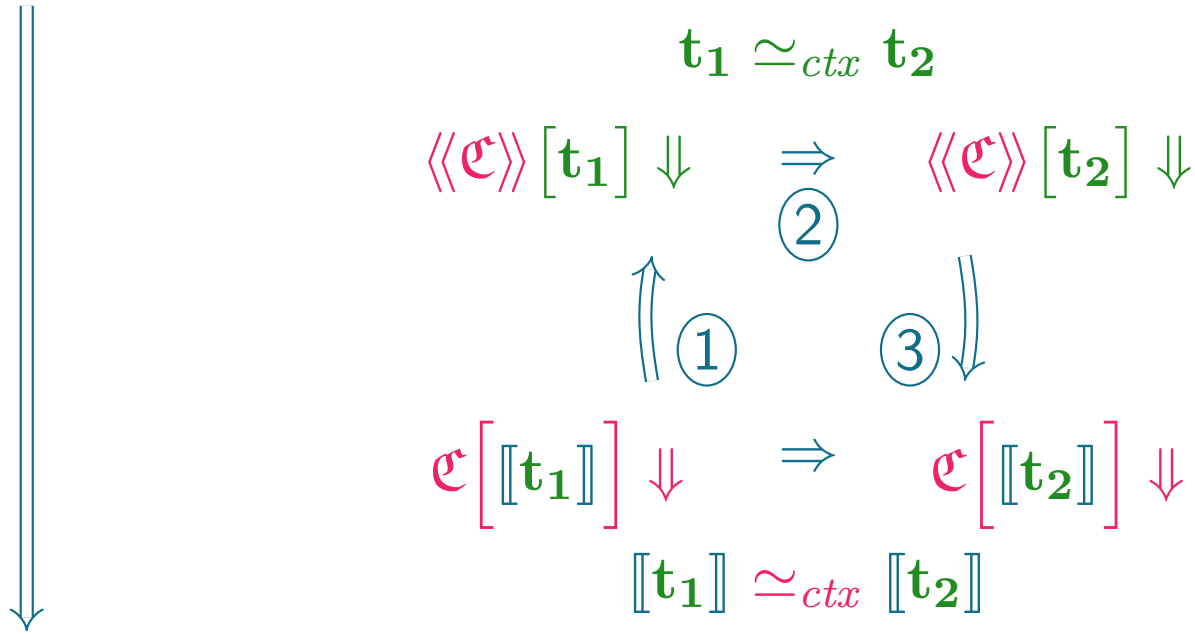
Back-translation of target contexts: $\langle\!\langle \mathfrak{C} \rangle\!\rangle \approx \mathfrak{C}$



KU LEUVEN

Contextual Equivalence Preservation (2/2).

Back-translation of target contexts: $\langle\!\langle \mathfrak{C} \rangle\!\rangle \approx \mathfrak{C}$



$$
\begin{aligned}
\mathbf{UVal} \stackrel{\text{def}}{=} \quad & \texttt{Bool} && \text{Booleans} \\
& \uplus \texttt{Unit} && \text{Unit value} \\
& \uplus (\mathbf{UVal} \times \mathbf{UVal}) && \text{Pairs} \\
& \uplus (\mathbf{UVal} \uplus \mathbf{UVal}) && \text{Sums} \\
& \uplus (\mathbf{UVal} \to \mathbf{UVal}) && \text{Functions}
\end{aligned}
$$

$$\langle\!\langle \mathfrak{C} \rangle\!\rangle_\tau = \langle\!\langle \mathfrak{C} \rangle\!\rangle [\mathbf{inject}_\tau \ \cdot]$$

$$\mathbf{inject}_\tau : \tau \to \mathbf{UVal}$$

$$\langle\!\langle \mathfrak{C} \rangle\!\rangle : \emptyset, \mathbf{UVal} \to \emptyset, \mathbf{UVal}$$

Problem: how to define $\mathbf{inject}_{\exists\alpha.\ldots}$?

$$
\begin{aligned}
\mathbf{UVal} \stackrel{\text{def}}{=} \quad & \texttt{Bool} && \text{Booleans} \\
& \uplus\, \texttt{Unit} && \text{Unit value} \\
& \uplus\, (\mathbf{UVal} \times \mathbf{UVal}) && \text{Pairs} \\
& \uplus\, (\mathbf{UVal} \uplus \mathbf{UVal}) && \text{Sums} \\
& \uplus\, (\mathbf{UVal} \to \mathbf{UVal}) && \text{Functions}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{UVal}_\tau \stackrel{\text{def}}{=} \quad & \texttt{Bool} && \text{Booleans} \\
\uplus\ & \texttt{Unit} && \text{Unit value} \\
\uplus\ & (\mathbf{UVal}_\tau \times \mathbf{UVal}_\tau) && \text{Pairs} \\
\uplus\ & (\mathbf{UVal}_\tau \uplus \mathbf{UVal}_\tau) && \text{Sums} \\
\uplus\ & (\mathbf{UVal}_\tau \to \mathbf{UVal}_\tau) && \text{Functions} \\
\uplus\ & \tau && \text{Type Vars}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{UVal}_\tau \stackrel{\text{def}}{=} \quad & \texttt{Bool} && \text{Booleans} \\
& \uplus\ \texttt{Unit} && \text{Unit value} \\
& \uplus\ (\mathbf{UVal}_\tau \times \mathbf{UVal}_\tau) && \text{Pairs} \\
& \uplus\ (\mathbf{UVal}_\tau \uplus \mathbf{UVal}_\tau) && \text{Sums} \\
& \uplus\ (\forall \delta'.\mathbf{UVal}_{\delta' \uplus \tau} \to \mathbf{UExpr}_{\delta' \uplus \tau}) && \text{Functions} \\
& \uplus\ \tau && \text{Type Vars}
\end{aligned}
$$

$$\mathbf{UExpr}_\tau \stackrel{\text{def}}{=} \exists \delta'.\, \mathbf{UVal}_{\delta' \uplus \tau}$$

$$
\begin{aligned}
\mathbf{UVal}_\tau \stackrel{\text{def}}{=} \quad & \texttt{Bool} && \text{Booleans} \\
& \uplus\ \texttt{Unit} && \text{Unit value} \\
& \uplus\ (\mathbf{UVal}_\tau \times \mathbf{UVal}_\tau) && \text{Pairs} \\
& \uplus\ (\mathbf{UVal}_\tau \uplus \mathbf{UVal}_\tau) && \text{Sums} \\
& \uplus\ (\forall \delta'.\mathbf{UVal}_{\delta' \uplus \tau} \to \mathbf{UExpr}_{\delta' \uplus \tau}) && \text{Functions} \\
& \uplus\ \tau && \text{Type Vars} \\
& \uplus\ (\tau \times (\tau \to \texttt{Bool})) && \text{Seal} \\
& \uplus\ (\tau \times \mathbf{UVal}_\tau) && \text{Sealed value}
\end{aligned}
$$

$$\mathbf{UExpr}_\tau \stackrel{\text{def}}{=} \exists \delta'.\, \mathbf{UVal}_{\delta' \uplus \tau}$$

$$
\begin{aligned}
\mathbf{UVal}_\tau \stackrel{\text{def}}{=} \quad & \texttt{Bool} && \text{Booleans} \\
\uplus\ & \texttt{Unit} && \text{Unit value} \\
\uplus\ & (\mathbf{UVal}_\tau \times \mathbf{UVal}_\tau) && \text{Pairs} \\
\uplus\ & (\mathbf{UVal}_\tau \uplus \mathbf{UVal}_\tau) && \text{Sums} \\
\uplus\ & (\forall \delta'.\mathbf{UVal}_{\delta' \uplus \tau} \to \mathbf{UExpr}_{\delta' \uplus \tau}) && \text{Functions} \\
\uplus\ & \tau && \text{Type Vars} \\
\uplus\ & (\tau \times (\tau \to \texttt{Bool})) && \text{Seal} \\
\uplus\ & (\tau \times \mathbf{UVal}_\tau) && \text{Sealed value}
\end{aligned}
$$

$$\mathbf{UExpr}_\tau \stackrel{\text{def}}{=} \exists \delta'.\, \mathbf{UVal}_{\delta' \uplus \tau}$$

Problem: we have $\mu\alpha.\cdots$, but this needs $\mu(\alpha :: * \to *).\cdots$.
Solution: Approximate!

- Full back-translation is not needed for full abstraction
- Approximate back-translation is sufficient
  - Fully accurate up to arbitrary n
  - Conservative beyond n
  - Prove using *directed*, *step-indexed* logical relations
- see Devriese, Patrignani, Piessens (POPL 2016)

Contextual Equivalence Preservation (2/2).

Approximate back-translation: $\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \gtrsim_n \mathfrak{C}$ $\quad \forall m.\, \langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \lesssim_m \mathfrak{C}$

$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$[\![\mathbf{t_1}]\!] \simeq_{ctx} [\![\mathbf{t_2}]\!]$$

Contextual Equivalence Preservation (2/2).

Approximate back-translation: $\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \gtrsim_n \mathfrak{C}$ $\quad$ $\forall m.\, \langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \lesssim_m \mathfrak{C}$

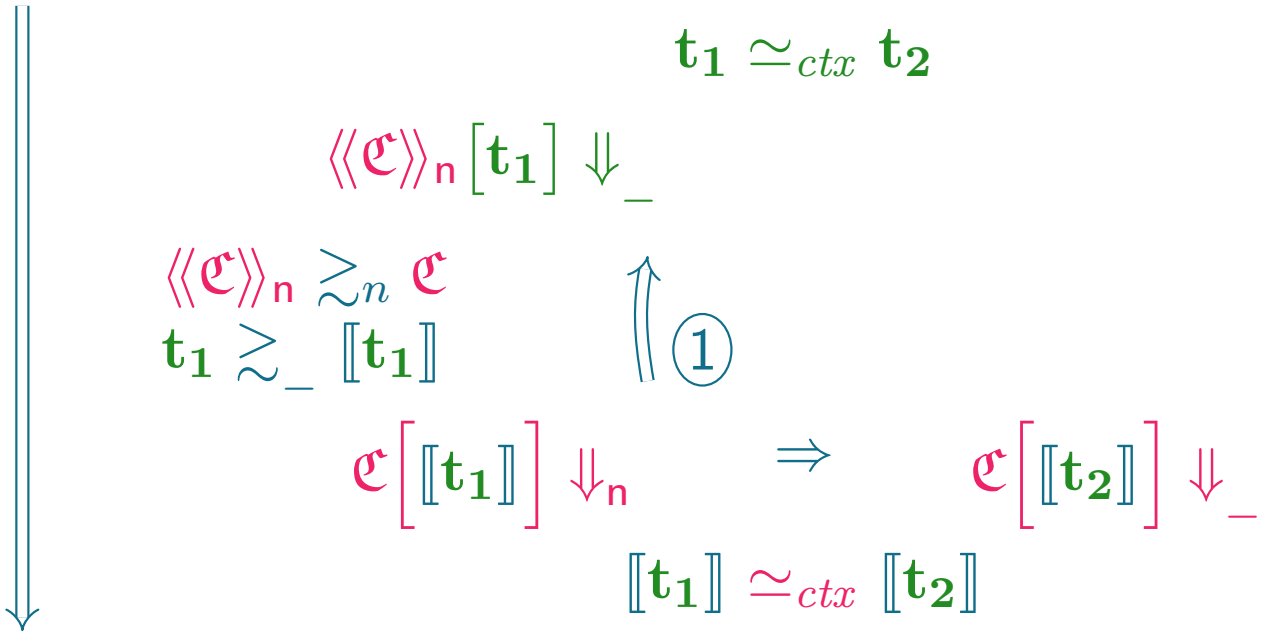
$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$\mathfrak{C}\left[\left[\mathbf{t_1}\right]\right] \Downarrow_{\mathsf{n}} \quad \Rightarrow \quad \mathfrak{C}\left[\left[\mathbf{t_2}\right]\right] \Downarrow_{\_}$$

$$\left[\left[\mathbf{t_1}\right]\right] \simeq_{ctx} \left[\left[\mathbf{t_2}\right]\right]$$

Contextual Equivalence Preservation (2/2).

Approximate back-translation: $\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \gtrsim_n \mathfrak{C}$ $\quad$ $\forall m.\ \langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \lesssim_m \mathfrak{C}$

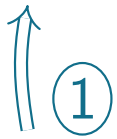


Contextual Equivalence Preservation (2/2).

Approximate back-translation: $\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \gtrsim_n \mathfrak{C}$ $\quad \forall m.\, \langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \lesssim_m \mathfrak{C}$

$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} [\mathbf{t_1}] \Downarrow_{\_} \quad \overset{②}{\Rightarrow} \quad \langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} [\mathbf{t_2}] \Downarrow_{\_}$$



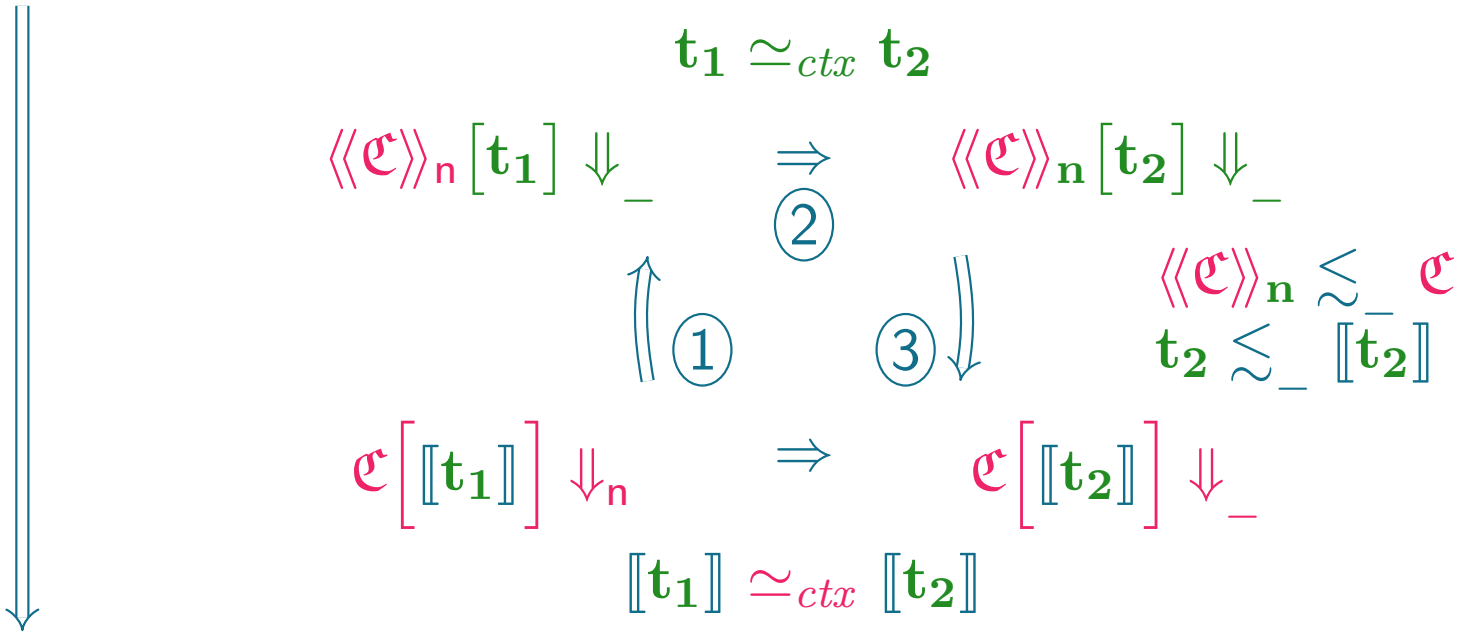
$$\mathfrak{C}[[\![\mathbf{t_1}]\!]] \Downarrow_{\mathsf{n}} \quad \Rightarrow \quad \mathfrak{C}[[\![\mathbf{t_2}]\!]] \Downarrow_{\_}$$

$$[\![\mathbf{t_1}]\!] \simeq_{ctx} [\![\mathbf{t_2}]\!]$$

KU LEUVEN

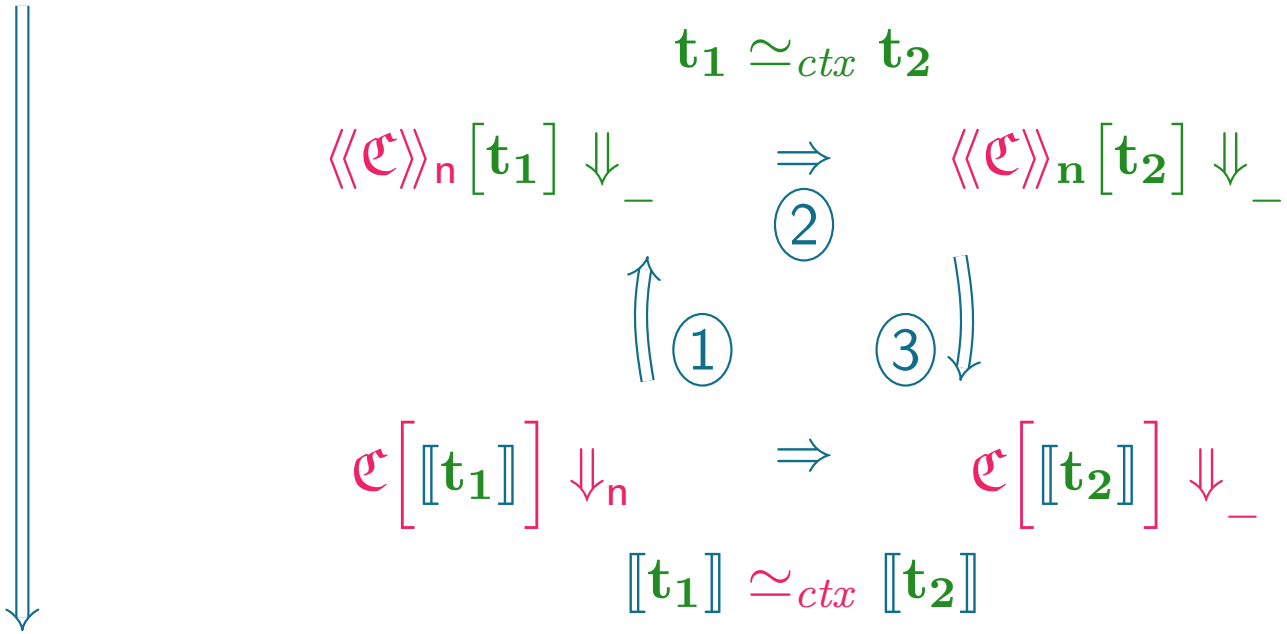
Contextual Equivalence Preservation (2/2).

Approximate back-translation: $\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \gtrsim_n \mathfrak{C}$ $\quad \forall m.\ \langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \lesssim_m \mathfrak{C}$



Contextual Equivalence Preservation (2/2).

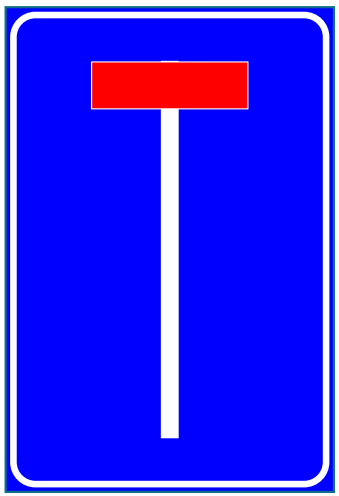
Approximate back-translation: $\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \gtrsim_n \mathfrak{C}$ $\quad \forall m.\, \langle\!\langle \mathfrak{C} \rangle\!\rangle_{\mathsf{n}} \lesssim_m \mathfrak{C}$



... 300 pages of tech report later...

... 300 pages of tech report later... a week before the POPL deadline ...

... 300 pages of tech report later... a week before the POPL deadline ...



It doesn't quite work...

- What type variables can an existential quantification close over?
- Difference between the following types?

$$\exists\alpha.\,\forall\beta.\,\beta \to (\beta \to \texttt{Bool}) \to \alpha \times (\alpha \to \texttt{Bool}) \qquad (1)$$

$$\forall\beta.\,\exists\alpha.\,\beta \to (\beta \to \texttt{Bool}) \to \alpha \times (\alpha \to \texttt{Bool}) \qquad (2)$$

- Our dynamic enforcement treats them identically!
- Back-translate a context $\lambda\_.\,\lambda\mathsf{x}.\,\lambda\mathsf{f}.\,\langle\mathsf{x},\mathsf{f}\rangle$?
- Should only work for (2) but not (1)?

Enforcing quantifier scope?

$$\exists\alpha.\,\forall\beta.\,\beta \to (\beta \to \texttt{Bool}) \to \alpha \times (\alpha \to \texttt{Bool}) \qquad (1)$$

vs.

$$\forall\beta.\,\exists\alpha.\,\beta \to (\beta \to \texttt{Bool}) \to \alpha \times (\alpha \to \texttt{Bool}) \qquad (2)$$

What to do?

- More complex back-translation possible? Construct closures somehow? What if existential variable appears less conveniently?
- Does full abstraction even hold for (1)? What if quantifier scope enforces some property (like in the ST monad)?
- The plan for now: prove full abstraction for a subset of types:
  - no $\forall$s inside $\exists$s
  - (2) is fine, but (1) isn't

- Questions?
- Feedback, suggestions?

Backup slide: the need for a type-indexed UVal

- How to back-translate $\mathsf{t} \stackrel{\text{def}}{=} \lambda \mathsf{x}.\, \mathsf{x}.1\ \mathsf{x}.2$ at type $(\exists\alpha.(\alpha \to \mathtt{Bool}) \times \alpha) \to \mathtt{Bool}$?
- When applied to $\mathrm{pack}\ \langle \mathtt{Bool}, \langle \lambda \mathbf{x} : \mathtt{Bool}.\, \mathbf{x}, \mathtt{true} \rangle \rangle\ \mathrm{as}\ \exists\alpha.(\alpha \to \mathtt{Bool}) \times \alpha$, back-translation should behave as t applied to $\langle \lambda \mathsf{x}.\, \mathsf{x}, \mathtt{true} \rangle$
- Only way to do that is to open up the pair:

$$\lambda x : \exists\alpha.(\alpha \to \mathtt{Bool}) \times \alpha.\, \mathrm{unpack}\ x\ \mathrm{as}\ \langle \alpha, x' \rangle\ \mathrm{in}\ x'.1\ x'.2$$

- What is the type of $\mathbf{x}'.\mathbf{2}$ here?
  - $\mathbf{UVal}_n$?
  - **Must** mention α!

Back-translation Proof Structure

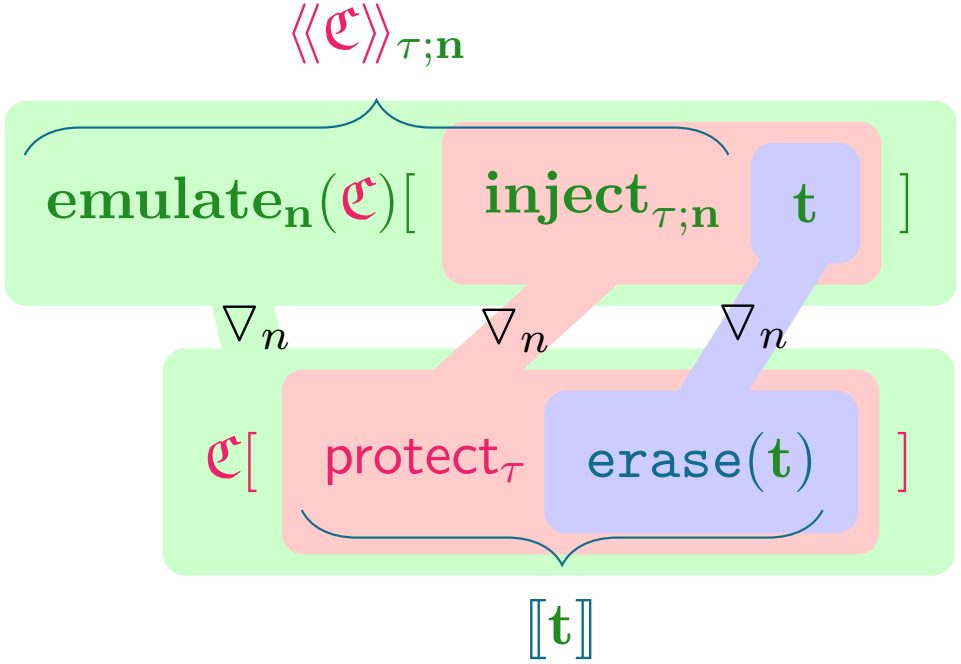
This statement

$\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\tau;\mathbf{n}}[\mathbf{t}]$

∇_n

$\mathfrak{C}[[\![\mathbf{t}]\!]]$

expands to this



Erasure Correctness (at type τ)

Protect/Confine $\approx$ Inject/Extract (at $\mathbf{EmulVal}_{\hat{\tau};\mathsf{n};\mathsf{p}}$)

Emulation Correctness (at $\mathbf{EmulVal}_{\mathbf{Empty};\mathsf{n};\mathsf{p}}$)

KU LEUVEN